Kotlin - Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. Kotlin is rich in built-in operators and provide the following types of operators:

- Arithmetic Operators
- Relational Operators
- Assignment Operators
- Unary Operators
- Logical Operators
- Bitwise Operations

Now let's look into these Kotlin Operators one by one.

(a) Kotlin Arithmetic Operators

Kotlin arithmetic operators are used to perform basic mathematical operations such as addition, subtraction, multiplication and division etc.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y

Example

Following example shows different calculations using Kotlin Arithmetic Operators:

```
fun main(args: Array<String>) {
   val x: Int = 40
   val y: Int = 20

   println("x + y = " +  (x + y))
   println("x - y = " +  (x - y))
   println("x / y = " +  (x / y))
   println("x * y = " +  (x * y))
```

```
    println("x % y = " +  (x % y))
}
```

When you run the above Kotlin program, it will generate the following output:

```
x + y = 60
x - y = 20
x / y = 2
x * y = 800
x % y = 0
```

(b) Kotlin Relational Operators

Kotlin relational (comparison) operators are used to compare two values, and returns a **Boolean** value: either **true** or **false**.

Operator	Name	Example
>	greater than	x > y
<	less than	x < y
>=	greater than or equal to	x >= y
<=	less than or equal to	x <= y
==	is equal to	x == y
!=	not equal to	x != y

Example

Following example shows different calculations using Kotlin Relational Operators:

```
fun main(args: Array<String>) {
    val x: Int = 40
    val y: Int = 20

    println("x > y = " +  (x > y))
    println("x < y = " +  (x < y))
    println("x >= y = " +  (x >= y))
    println("x <= y = " +  (x <= y))
    println("x == y = " +  (x == y))
    println("x != y = " +  (x != y))
}
```

When you run the above Kotlin program, it will generate the following output:

```
x > y = true
x < y = false
x >= y = true
x <= y = false
x == y = false
x != y = true
```

(c) Kotlin Assignment Operators

Kotlin assignment operators are used to assign values to variables.

Following is an example where we used assignment operator **=** to assign a values into two variables:

```
fun main(args: Array<String>) {
   val x: Int = 40
   val y: Int = 20

   println("x = " +  x)
   println("y = " +  y)
}
```

When you run the above Kotlin program, it will generate the following output:

```
x = 40
y = 20
```

Following is one more example where we used assignment operator **+=** to add the value of self variable and assign it back into the same variable:

```
fun main(args: Array<String>) {
   var x: Int = 40

   x += 10

   println("x = " +  x)
}
```

When you run the above Kotlin program, it will generate the following output:

```
x = 50
```

Following is a list of all assignment operators:

Operator	Example	Expanded Form
=	x = 10	x = 10
+=	x += 10	x = x - 10
-=	x -= 10	x = x - 10
*=	x *= 10	x = x * 10
/=	x /= 10	x = x / 10
%=	x %= 10	x = x % 10

Example

Following example shows different calculations using Kotlin Assignment Operators:

```
fun main(args: Array<String>) {
   var x: Int = 40

   x += 5
   println("x += 5 = " + x )

   x = 40;
   x -= 5
   println("x -= 5 = " +  x)

   x = 40
   x *= 5
   println("x *= 5 = " +  x)

   x = 40
   x /= 5
   println("x /= 5 = " +  x)

   x = 43
   x %= 5
   println("x %= 5 = " + x)
}
```

When you run the above Kotlin program, it will generate the following output:

```
x += 5 = 45
x -= 5 = 35
x *= 5 = 200
x /= 5 = 8
x %= 5 = 3
```

(d) Kotlin Unary Operators

The unary operators require only one operand; they perform various operations such as incrementing/decrementing a value by one, negating an expression, or inverting the value of a boolean.

Following is the list of Kotlin Unary Operators:

Operator	Name	Example
+	unary plus	+x
-	unary minus	-x
++	increment by 1	++x
--	decrement by 1	--x
!	inverts the value of a boolean	!x

Example

Following example shows different calculations using Kotlin Unary Operators:

```
fun main(args: Array<String>) {
   var x: Int = 40
   var b:Boolean = true

   println("+x = " +  (+x))
   println("-x = " +  (-x))
   println("++x = " +  (++x))
   println("--x = " +  (--x))
   println("!b = " +  (!b))
}
```

When you run the above Kotlin program, it will generate the following output:

```
+x = 40
-x = -40
++x = 41
--x = 40
!b = false
```

Here increment (++) and decrement (--) operators can be used as prefix as ++x or --x as well as suffix as x++ or x--. The only difference between the two forms is that in case we use them as prefix then operator will apply before expression is executed, but if use them as suffix then operator will apply after the expression is executed.

(e) Kotlin Logical Operators

Kotlin logical operators are used to determine the logic between two variables or values:

Following is the list of Kotlin Logical Operators:

Operator	Name	Description	Example
&&	Logical and	Returns true if both operands are true	x && y
||	Logical or	Returns true if either of the operands is true	x || y
!	Logical not	Reverse the result, returns false if the operand is true	!x

Example

Following example shows different calculations using Kotlin Logical Operators:

```
fun main(args: Array<String>) {
   var x: Boolean = true
   var y:Boolean = false

   println("x && y = " +  (x && y))
   println("x || y = " +  (x || y))
   println("!y = " +  (!y))
}
```

When you run the above Kotlin program, it will generate the following output:

```
x && y = false
x || y = true
!y = true
```

(e) Kotlin Bitwise Operations

Kotlin does not have any bitwise operators but Kotlin provides a list of helper functions to perform bitwise operations.

Following is the list of Kotlin Bitwise Functions:

Function	Description	Example
shl (bits)	signed shift left	x.shl(y)
shr (bits)	signed shift right	x.shr(y)
ushr (bits)	unsigned shift right	x.ushr(y)
and (bits)	bitwise and	x.and(y)
or (bits)	bitwise or	x.or(y)
xor (bits)	bitwise xor	x.xor(y)
inv()	bitwise inverse	x.inv()

Example

Following example shows different calculations using Kotlin bitwise functions:

```
fun main(args: Array<String>) {
   var x:Int = 60     // 60 = 0011 1100
   var y:Int = 13     // 13 = 0000 1101
   var z:Int

   z = x.shl(2)       // 240 = 1111 0000
   println("x.shl(2) = " +  z)

   z = x.shr(2)       // 15 = 0000 1111
   println("x.shr(2) = " +  z)

   z = x.and(y)       // 12 = 0000 1100
   println("x.and(y)  = " +  z)

   z = x.or(y)        // 61 = 0011 1101
   println("x.or(y)  = " +  z)

   z = x.xor(y)       // 49 = 0011 0001
   println("x.xor(y)  = " +  z)

   z = x.inv()        // -61 = 1100 0011
   println("x.inv()  = " +  z)
}
```

When you run the above Kotlin program, it will generate the following output:

```
x.shl(2) = 240
x.shr(2) = 15
x.and(y) = 12
x.or(y) = 61
x.xor(y) = 49
x.inv() = -61
```

Quiz Time (Interview & Exams Preparation)

Q 1 - What does the Kotlin operator % do?

A - It is used to divide a number by another number.

B - Kotlin does not support any such operator

C - This is bitwise XOR operator

D - This is called modulus operator and returns the division remainder.

Q 2 - Kotlin supports a good number of bitwise operators

A - Correct

B - Incorrect

Q 3 - What does Kotlin operator ++ do?

A - It is used to add two values

B - There is no any such operators like ++ in Kotlin

C - This is called unary increment operator

D - None of the above

Q 4 - Which of the following function will do bitwise right shift operation?

A - x.ushr(y)

B - x.shr(y)

C - x.shl(y)

D - None of the above

Q 5 - Which of the following is a logical inverse operator:

A - inv()

B - !

C - &&

D - ||

Q 6 - What will be the output of the following Kotlin code:

```
fun main(args: Array<String>) {
   var x: Int = 40
```

```
    x += 10

    println(x)
}
```

A - 40

B - Syntax Error

C - 50

D None of the above

Q 7 - What will be the output of the following Kotlin code:

```
fun main(args: Array<String>) {
    var x: Int = 60

    println(x.shr(2))
}
```

A - 15

B - Syntax Error

C - 50

D None of the above